8. Window Design

8.1 Primary Windows

A primary window, shown in figure 8-1, includes a window title in the title bar and may include a menu bar, toolbar, and status bar in the client area. The window has a horizontal and/or vertical scroll bar if the client area is too small to view all of the contents of the window. When scroll bars are included, they scroll the main part of the window and not the menu bar or status bar. If desired, a primary window can be divided into paned areas when space is limited or to present two simultaneous views of the same data in a single window. Panes can be arranged either horizontally or vertically and can be either a fixed size or adjustable by users. If users can adjust pane size, the window includes a sash (in Motif) or a split box (in MS Windows).

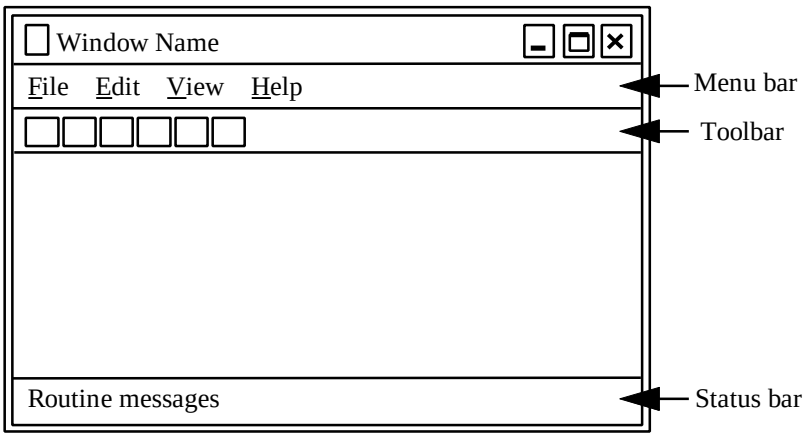

Figure 8-1. Example primary window in MS Windows.

If a primary window presents controls, they are arranged as described in section 8.2.2. Push buttons can be included in a primary window in order to provide access to actions that are chosen frequently (see section 8.2.3). A primary window can make use of tabbed pages (see section 6.7) instead of scroll bars to organize groups of controls that need not all be displayed at the same time.

8.1.1 Window Title

Motif Only: The title of a primary window consists of a text label centered in the title bar.

MS Windows Only: The title of a primary window consists of a title bar icon and text label placed at the left edge of the title bar. The small version of the application icon is used if the window represents an application that does not create its own files or documents. The icon for the appropriate file or document type is used if the window provides a view of one of these files or documents.

If the application has a single primary window, the window title is the name of the application. If the application has multiple primary windows, each is identified by application and task name.

Motif Only: If the application is used to view the contents of a file, the window title includes the application name followed by the file name.

MS Windows Only: If the application is used to view the contents of a file, the window title includes the file name and may include the application name. If the application name is used, the file name is presented first, followed by a hyphen and the application name.

MS Windows Only: The title of the MDI parent application window (see section 7.1.2.3) consists of the application icon, followed by the application name. The title of an MDI child document window consists of the appropriate file or document icon, followed by the name of the document. If a document window is maximized, the title of the parent window contains the application icon followed by the application name and the document name; the document icon is placed at the left edge of the menu bar, preceding the title of the File menu.

The first letter of each word in the window title is capitalized, except for prepositions and articles. The window title does not contain information such as the version of an application or the full path name for a file. If users open an application that creates a new (i.e., unnamed) file, the window is assigned a default name (e.g., Untitled). If users open more than one instance of a window with a given name, each instance is numbered (e.g., Untitled 1, Untitled 2, etc.). If the name of the object being viewed in the window changes (e.g., the user renames the object), the window title is also updated.

8.1.2 Menu Bar

The menu bar is placed below the title bar at the top of the client area and contains no more than ten menu titles plus Help. The space between menu titles is sufficient (at least three character widths) so that multi-word titles can be distinguished from single-word titles. Commands (e.g., push buttons) are not included in the menu bar. If a window contains tabbed pages, the contents of the menu bar can change as needed to apply to the page currently in the foreground.

Motif Only: If the menu bar contains any of the following common menus, they are ordered: File, Edit, View, Options, Help. The menu titles begin at the left margin of the menu bar and extend rightward, with Help at the right margin of the menu bar.

MS Windows Only: If the menu bar contains any of the following common menus, they are ordered: File, Edit, View, Window, Help. The menu titles begin at the left margin of the menu bar and extend rightward, with Help placed next to the menu that precedes it.

Application-specific menus can be inserted between these common ones, except as indicated below. Menu design guidelines are presented in section 5.5.

File menu. The first (i.e., leftmost) menu in the menu bar contains options for users to work with the data in the window as a whole. The title of this menu is File if the window uses a document

metaphor and directly interacts with files. If the title of this menu is not File, it is an application-specific term that describes the type of object(s) the window operates on.

Motif Only: If the File menu includes any of the following options, they are ordered: New, Open, Save, Save As, Print, Close, and Exit. Separators follow the Open, Save As, and Print options.

MS Windows Only: If the File menu includes any of the following options, they are ordered: New, Open, Close, Save, Save As, Page Setup, Print, and Exit. Separators follow the Close, Save As, and Print options.[1]

MS Windows Only: If the File menu includes options listing the most recently used files, the options precede Exit. Selecting one of these options opens a window containing the file. If the file is already open, selecting the option raises that window to the foreground. The file list can include up to eight options. When a file is opened, its name (e.g., 1 test.doc) is placed at the top of the list and it is given the number 1 that serves as its mnemonic. When another file is opened, it is placed to the top of the list (and given the number 1), and the previously opened files are renumbered and move down in the list.

Edit menu. The Edit menu contains options that enable users to modify the data in the window. If File and Edit are both included, they are next to each other in the menu bar.

Motif Only: If the Edit menu includes any of the following options, they are ordered: Undo, Cut, Copy, Paste, Clear, Delete, Select All, Deselect All, and Properties. Separators follow the Undo, Paste, and Delete options.

MS Windows Only: If the Edit menu includes any of the following options, they are ordered: Undo, Repeat, Cut, Copy, Paste, Paste Special, Clear, Delete, Select All, Find, and Replace. Separators follow the Repeat, Paste Special, and Select All options.

View menu. The View menu contains options for changing the user's view of an object but does not actually change the object. This menu can also contain options for controlling the display of window components such as a toolbar or status bar.

Options menu. The Options menu contains options for customizing the application.

Window menu. The Window menu is included in the parent window in an MDI application (see section 7.1.2.3) and contains options for arranging child document windows within the parent (e.g., Tile, Cascade) and for closing these windows. This menu also includes options listing the names of all child windows; selecting one of these options activates the window and raises it to the foreground.

[1] The current version of the MS Windows style guide does not specify the contents of the File and Edit menus. The specifications provided here are based on implementations in COTS products such as Microsoft Office.

Help menu. The Help menu provides access to additional information about the window or the application. Section 11 provides additional information on user support resources.

Motif Only: The Help menu includes On Item and On AppName options and may provide access to support documentation as defined by the application.

MS Windows Only: The Help menu includes an On AppName option and may provide access to support documentation as defined by the application.

8.1.3 Toolbar

A toolbar is a collection of buttons that can invoke modes (e.g., different drawing tools), apply settings (e.g., select font style or size), and execute actions (e.g., invoke a print command) in a window. A toolbar provides redundant access to functionality available elsewhere in the window (e.g., in a pull-down menu). A toolbar can occupy a fixed position within a window (e.g., below the menu bar), or its location can be configurable by users; a toolbar can also be placed in a separate dialog window. Users have the option to hide or show the toolbar (e.g., from a toggle-type option in the View menu). If desired, users can also be allowed to change or rearrange the contents of the toolbar and to choose the size of toolbar buttons.

All of the buttons in a toolbar are the same size. If the toolbar includes any of the following actions, they are ordered: New, Open, Save, Print, Cut, Copy, Paste, and Help. Any application-specific actions are inserted after Paste. The buttons can be evenly spaced across the toolbar or placed in groups (e.g., buttons for New, Open, Save, Print in one group, buttons for Cut, Copy, Paste in another). Application-specific actions are arranged in an order expected by users; if one does not exist, they are arranged by frequency of occurrence, sequence of use, or importance.

The normal appearance of a toolbar button is unselected; the button appearance changes (i.e., highlights) when it is selected, and is grayed out when it is unavailable for selection. The appearance of a toolbar button is linked to that of the menu option or control for which it provides redundant access; for example, whenever the menu option becomes unavailable, the corresponding toolbar button is also shown as unavailable.

The buttons in the toolbar contain graphic labels representing the action executed when the button is selected. The graphic is centered on the face of the button. If the graphic is supplemented with a text label, the graphic is placed either above the text or to the left of it.

MS Windows Only: The application provides tooltips (see user support in section 11) for toolbar buttons that do not have text labels.

If the application implements toolbar buttons for any of the actions listed in appendix D, it uses the graphics in this appendix. If application-specific graphics are created, they represent a verb (rather than a noun) and can depict a before-and-after representation of the action (e.g., a small and large version of an object, connected by an arrow, to indicate Magnify), the tool that would accomplish the action (e.g., a pair paperclip to indicate Attach), or the action itself (e.g., a paintbrush filling an object with color to indicate Paint). One of these schemes is selected, and all

of the graphics created by the application are designed to fit this scheme. The application does not use a triangular arrow graphic in a toolbar button since Motif and MS Windows use this shape to indicate when a control will display additional information.

When users choose a toolbar button that controls a property of an object (e.g., the style of font), the property is immediately applied to the current selection (e.g., the text that is highlighted); users do not have to activate a push button in the window to invoke the property. When users choose a toolbar button that invokes a mode, the button remains selected (i.e., highlighted) as long as the mode is in effect. The pointer shape changes to indicate type of operation users can perform while in the mode. The pointer has this shape whenever it is in the window where the mode is in effect. If users move the pointer outside this window, the pointer changes to the appropriate shape. The toolbar either provides a button for returning to an unselected state (i.e., exiting the mode) or automatically returns to an unselected state after an action is executed.

8.1.4 Status Bar

A status bar is placed at the bottom of the client area and extends the width of the window. The status bar can be tall enough to display up to five lines of text and include a scroll bar if the information being presented exceeds the space available. The status bar normally displays read-only text, either as a label or in a text-display box. The status bar is separated from the other window contents by empty space or a separator that extends the width of the window. If a status bar is used in a window, users have the option to show or hide it (e.g., from a toggle-type option in the View menu).

The status bar is used to present noncritical information to users, provide simple help, and indicate the status or current state of the application (e.g., cursor location, modes in effect). The status bar can be divided into parts to display more than one type of status information; if it is divided into parts, the text in each part is left-aligned.

The information in the status bar is removed as soon as it is no longer relevant to the current state of the window or the current object with focus. For example, when the status bar is used to indicate the status of an action, a progress message is displayed when the action is initiated (e.g., “Drawing map...”), updated when the action is completed (e.g., “Drawing map...Done”), and removed within 5 sec. of action completion.

8.2 Secondary Windows

A secondary window, shown in figure 8-2, includes a window title in the title bar and control and action areas in the client portion of the window. A dialog window may also include a menu bar and status bar in the client area. If the window has a status bar, it is located at the bottom of the client area and the action area is placed above it. A dialog window can include a menu bar if the number of actions executed in the window is more than six and/or if access to File and Edit operations is needed. If a dialog window includes a menu bar or a status bar, the application follows the design guidelines for these components described in section 8.1.

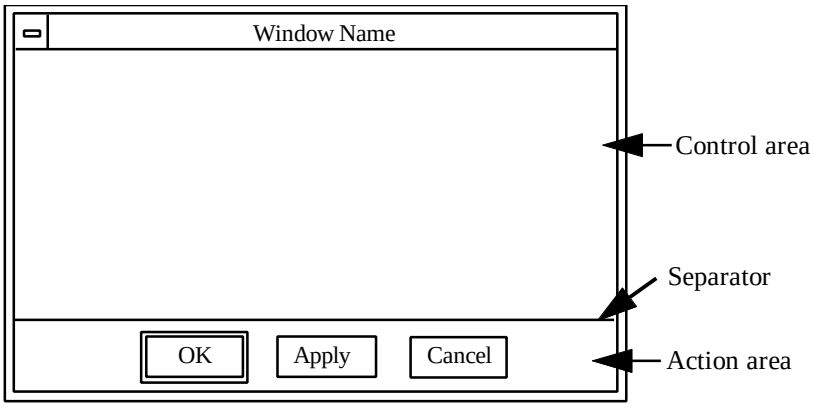

Figure 8-2. Example secondary window in Motif.

Section 9 defines the format for dialog and message windows that perform common functions. When the application supports one of these functions, it does so using the format indicated in that section. When the application supports its unique mission functions, the dialog and message windows it displays follow the design guidelines described below.

8.2.1 Window Title

Motif Only: The title of a secondary window is a text label centered in the title bar. The title identifies the type or purpose of the window and may include the application name as part of the title.

MS Windows Only: The title of a secondary window is a text label placed at the left edge of the title bar; the title does not include a title bar icon. The title of a dialog window describes the type or purpose of the window; the title of a message window identifies the object to which the message applies.

The first letter of each word in the window title is capitalized, except for prepositions and articles. The window title is not used to present dynamic information such as messages to the user. If selecting a menu option causes a dialog window to be displayed, the window title matches or refers to the wording of the option that displayed it.

8.2.2 Control Area

8.2.2.1 Arrangement of Controls

Group boxes. A group box identifies a set of related controls and distinguishes them from other controls in the window. A group box is normally used with radio buttons, check boxes, and text boxes (see figure 8-3) and not used with push buttons, a single control (e.g., a list box), or in a window with only one tab group. When a set of controls is defined as a tab group, all of the controls are placed inside the group box. The box has an “etched in,” rather than a “shadow in” or “shadow out,” effect.

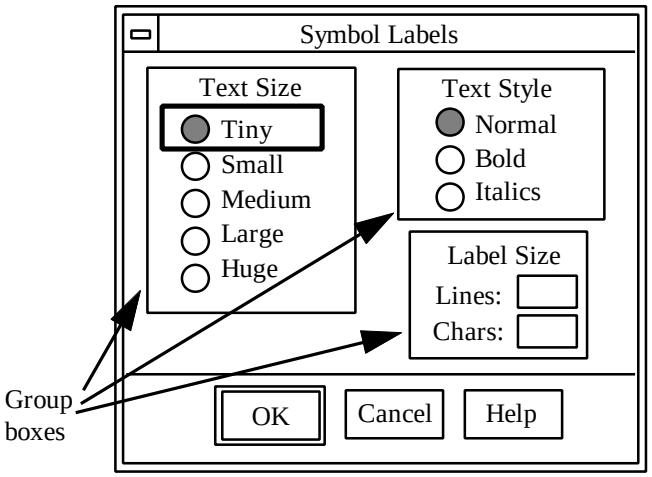

Figure 8-3. Example group boxes in Motif.

If a group box includes a heading, it is a label describing the function performed by the controls, and it is not followed by a colon. The first letter of each word in the heading is capitalized, except for prepositions and articles.[2]

Motif Only: The heading is placed at the top of the group box and either centered or left-aligned with the other controls in the box.

MS Windows Only: The heading is placed in the group box frame in the upper left corner and left aligned with the other controls in the box.

If the heading is longer than the width of the controls in the group box, the size of the box is extended so that it is wider than the heading.

Radio buttons and check boxes. Radio buttons or check boxes are arranged in one or more rows or columns within a group box. The preferred arrangement is vertical and left-aligned; if placed horizontally, space is sufficient (at least twice the distance between the button/box indicator and its label) so the indicator is paired with the label on the right, not the left.

Text boxes. Text boxes are placed together in a group box based on sequence of use, frequency of use, or importance. If the labels and text boxes are placed adjacent to each other, the labels and boxes are both left-aligned, or the labels are right-aligned and the text boxes left-aligned, as shown in figure 8-4. If the label is placed above the text box, the label is aligned with the left end of the box.

[2] MIL-STD 1472E indicates that the heading shall be displayed in upper-case letters.

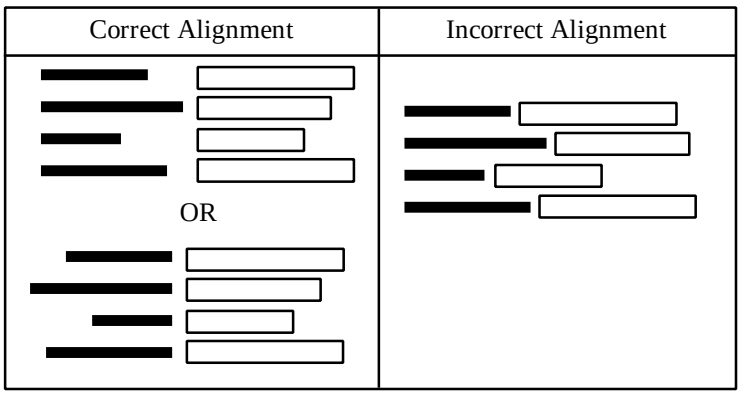

Figure 8-4. Correct and incorrect alignment of labels and text boxes.

A conditional (or dependent) text box is placed to the right of or below the control to which it relates. The text box can be either unavailable (with its label grayed out) or not displayed at all until the control to which it relates is selected.

The label for each text box in a group is worded to be clearly different from the other text box labels in the group. If the labels are highly redundant (e.g., ship name, ship UIC, ship homeport), the common word is removed from the label (e.g., Name, UIC, Homeport) and placed instead in the heading for the group box (e.g., Ship Information).

If the information being entered is tabular, the text boxes are arranged in rows and/or columns, with each one labeled. If the text entry area is scrollable, the row and column headings are placed outside the scrollable area so that they remain visible when scrolling occurs. If users are entering data from a hardcopy form, the window format is identical to the hardcopy format.

Mnemonics (MS Windows Only).[3] A mnemonic is included in the label for each text box, in each push button with a text label, and in each radio button and check box in a tab group. If a label is added to a control (e.g., in text boxes and list boxes), activating the mnemonic in the label assigns keyboard focus to the control, not the label. If the application maps RETURN and ESC to the OK and Cancel push buttons, it does not define mnemonics for these buttons. Within a given window, the mnemonics in the menu bar (if one is present) and the controls in the client area are unique.

Default control settings. When a window is opened, all of the controls in the window reflect the current state of the application. For example, a window that allows users to change the font size of text in the application is displayed with the current font selected. If there is a preferred or expected choice within a window, a default option is defined within a control (e.g., a list box) or group of controls (e.g., a set of check boxes), and this choice is selected (i.e., highlighted) when the window is opened. If the expected choice cannot be anticipated, then a default is not designated. Controls that become unavailable are dimmed and not available for selection.

[3] The availability of mnemonics for the controls in a window is considered optional in Motif. If the application implements them, it does so according to the specifications defined here.

Controls that are never available to users (e.g., if access to them is password controlled) are not included in the window.

8.2.2.2 Navigation Among Controls

The application defines tab groups based on the order in which users are expected to interact with the controls in a window. This sequence is left to right, top to bottom in the window. For example, in figure 8-3, users tab from the radio button group in Text Size to the group in Text Style, then to each of the text fields in Label Size, and finally to the push button group. Text boxes are considered as individual tab groups, with TAB used for navigation between groups, while push buttons can be defined as individual tab groups or treated as a single tab group.[4]

When a window is opened, the location cursor is placed on the control with which users are expected to interact first. This control is usually the top leftmost one on which the location cursor can be positioned. For example, when the window in figure 8-3 is displayed, the location cursor is placed on the Tiny radio button in the Text Size tab group. When a window regains input focus, the location cursor is placed on the control that last had focus, provided that the control remains available for selection; otherwise, the location cursor is placed on the control users are most likely to select in the window.

The direction of cursor movement within a window is from upper left to lower right unless the object is scrollable; the location cursor wraps between the first and last tab groups in the window. When the location cursor moves into a tab group, the cursor is placed on the first available control in the group. The location cursor skips a tab group if none of the controls can have keyboard focus.

> Motif Only: In a scrollable control, keyboard focus remains on the element where it was positioned before the scrolling operation began even though the location cursor may no longer be in view. However, any keyboard action that moves the location cursor or makes a modification at the cursor location scrolls the control so that the cursor is visible.
>
> MS Windows Only: In a scrollable control, the location cursor moves when users scroll in page increments so that the cursor remains visible.

Once the location cursor is positioned within a tab group, the arrow keys move the cursor between the available controls within the group. For example, in the Text Size tab group in figure 8-3, DOWN moves the location cursor from Tiny to Small to Medium and so on. The location cursor is always visible as it moves within a window; i.e., there are no “invisible” tab groups that cause the location cursor to temporarily disappear as users navigate in the window. The position of the location cursor is not affected by movement of the pointer within the window; for example, the location cursor does not appear on a control within a tab group when the pointer moves into the group, and then disappear when the pointer moves out of the group. In addition, moving the location cursor to a control does not alter the size or position of the control.

[4] In previous versions of Motif, push buttons were defined as a single tab group; CDE provides the flexibility to use either TAB or the arrow keys to navigate within a push button group.

8.2.3 Action Area

Push buttons related to overall window functionality are placed in the action area of the window. In Motif, this area is along the bottom of the window and delimited from the control area with a separator, as shown in figure 8-2. In MS Windows, this area is either along the bottom of the window or down the right side of the window. The preferred implementation in DII applications is placement along the bottom of the window.[5] If users can perform multiple actions within a window and these actions can affect different objects in the window, the push buttons are placed near the object(s) to which they relate and labeled to identify the object(s) that each button affects.

Motif Only: Push buttons are displayed horizontally and centered in the action area, as shown in figure 8-2. Positive actions are placed first in the push button group, followed by buttons indicating negative actions and canceling actions (e.g., Cancel, Close), and then a Help push button (if one is included in the window).[6] If a default action is available in the window, it is the leftmost push button in the action area. If the window supports mutually exclusive actions (e.g., Pause and Resume), it contains separate push buttons for each action, and the label of the button that is not available at the time is grayed out.

MS Windows Only: If push buttons are displayed horizontally in the action area, they are placed at either the left or right margin of the window, as shown in figure 8-5. If there is an OK button, it is placed first (even if it is not the default), followed by Cancel, and then the other buttons in the group and Help (if one is included in the window). If a window supports mutually exclusive actions, it contains a single push button for these actions, with the label changing to indicate the action available.

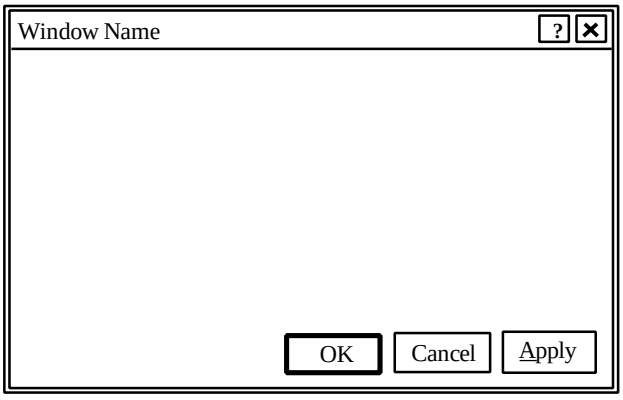

Figure 8-5. Example push button placement in MS Windows.

When a window opens, the default push button (if one is defined) is one that is performs a nondestructive action. The default designation may move to a different push button depending on

[5] This implementation provides consistency in visual design for dialog windows that can be implemented in both Motif and MS Windows applications.

[6] Motif requires a Help push button to be included in all dialog windows; MS Windows considers a Help button to be optional. The specifications provided here call for this button to be optional.

the control that has keyboard focus in the window. When keyboard focus is on a push button, its action is the default, and it is shown with default highlighting. This highlight moves with the location cursor during keyboard navigation in a group of push buttons and returns to the original button when focus leaves the push button group. If the default action in a window varies, one push button always shows the default highlighting except when there is no default action currently available.

8.2.4 Expandable Dialog Windows

The application can increase the size of a dialog window by including an Unfold push button in the window. When users select this button, the window expands to display additional information and/or actions. For example, a Print dialog window in Motif includes a common area for entering print parameters, with information about application-specific functions visible in an optional area when users activate a More button in the window. The button label includes ">>" (e.g., More >>) to indicate that it performs this action. Clicking BLeft on an Unfold button expands the window to its larger size. Clicking BLeft on the button a second time returns the window to its original size.

8.2.5 Text and Alerting in Message Windows

If a message window includes a text message, it uses language that is meaningful to users and requires no further documentation or translation. The text is left aligned within the window. When a message contains more than one sentence, the important information is placed at the start of the message. The text is worded so that the action users are asked to perform can appear as a push button in the window. For example, a window displaying the message "Confirm deletion of file" contains Delete and Cancel push buttons. Except for a Working window that closes when processing is done, the application does not present timed-information message windows (i.e., windows that present information for a fixed time period) and then resume processing without requiring a user response.

When a message window containing critical information is displayed, it can be accompanied by auditory feedback (e.g., a beep) as a secondary indicator to attract the user's attention. If the application provides this form of feedback, users can set auditory signals at a very low intensity or disable them as required (e.g., for rig-for-quiet operations on submarines).

8.3 Considerations in Window Design

The guidelines in this section are provided to assist developers in making decisions concerning effective window design and are not considered in determining DII style compliance.

8.3.1 Selecting Interface Components to Match User Actions

The application matches the interface components in a window to the actions that users are expected to execute in the window. The following are guidelines on how to choose the most appropriate components:

- Use radio buttons, an option menu, or a list box when users need to select from discrete values or choices; use a scale or spin box when users need to select from a continuous range of values.

- Use radio buttons when users need to see all of the settings available in a group; use an option menu or drop-down list box when users need to see only the current choice.

- Use radio buttons or an option menu when the set of options from which users choose is not likely to change; use a list box when the options from which users choose may change.

- When users have to make a single selection, use a group of radio buttons if there are up to seven choices; use an option menu if there are up to 12 choices; and use a list box or drop-down list box if there are more than 12 choices.

- When users have to make multiple selections, use check boxes if there are up to seven choices; use a list box otherwise.

- Use push buttons for frequently executed actions when space is available to display the buttons; use tear-off menus for these actions if space is limited.

- Use option menus when users need to set values or choose from a set of related options; use push buttons when users need to activate commands.

- Use a scale with a slider when users need to enter approximate values; use a spin box when users need to make precise entries.

8.3.2 Arranging Components by Importance and Scanning Order

The application places the most important information and components in the upper left part of the window, working down to the less important ones at the bottom of the window. The information is arranged to accommodate the possibility that users will resize the window. Placing the most important information in the upper-left corner of the window allows users to manipulate it even if the window is reduced to a size that is smaller than normal. The application visually sets apart task-critical information from other information in the window so that it can be easily seen. The window is designed from the perspective of what is logical to users and appropriate to the actions being executed, not what is logical or appealing to the developer.

8.3.3 Designing for Efficiency in Task Performance

The application arranges the components in a window so that users can move quickly and easily among them. The amount of pointer movement and/or the number of keystrokes required to perform a task is minimized. Likewise, components are arranged to minimize the amount of hand movement between the keyboard and pointing device when users work in a window; users are not forced to change input devices repeatedly when performing a task.

The application minimizes the number of actions users have to execute in order to complete a task. For example, the application prefills text entry boxes (e.g., current date, ownship name,

position) with default data values whenever possible. Similarly, if the application repeats the same text boxes in multiple windows, users have to enter the data once, with the application automatically placing this value in the text box as the default whenever it recurs.

The same window design is employed whenever users have to perform the same basic task; for example, a single design is used to present identifying information on data records, whether the data are tracks or messages. Different or distinctive elements can appear in a window to fit the task being performed, but the placement of these elements is consistent across windows within the application.

The application is designed so that each window contains all of the information users need in order to perform a task and users can complete the task without having to refer to information not included in the window. The application should not be structured so that it presents users with a sequence of individual windows, each of which contains only a portion of the overall task being performed. While this type of design may be efficient from a software development perspective (e.g., by allowing re-use of previously designed windows), it can produce an unnecessarily complex task navigation problem for users, increase the difficulty of the task being performed, and place an unnecessary memory load on them to remember information not included in the window.

8.3.4 Minimizing the Possibility of User Error

The application is designed to minimize the possibility of user error. The application provides context-sensitive visual and behavioral cues that encourage users to perform legal operations and that prevent them from performing illegal or incorrect operations, rather than relying on error messages to inform users after an error has been made. Only those actions that are relevant are available to the user. Components that cannot be selected are visually deemphasized by graying out to indicate their unavailability.

The actions available for each component in a window match the action(s) users are expected to perform with the component. For example, if users are required to select a single item in a list box, the available selection methods are limited to this type of action. Users are not allowed to select multiple items in the list, after which they are informed in an error message that they should have selected only one item.

Efforts to minimize the possibility of user error do not constrain users unnecessarily as they work in the application. For example, a window containing several mandatory text boxes disables the OK or Save push buttons until these boxes are filled so that users are unable to save the data before completing the mandatory entries. This approach allows users to fill the text boxes in whatever order they choose or to make corrections as desired before attempting to save the data. The application does not require users to enter a value before being allowed to leave the box. This approach forces users to perform data entry in a lock-step sequence defined by the application and restricts their ability to perform the task in a manner that does not match this sequence. The application needs to be designed to provide flexibility and support the user's sense of control, while at the same time bounding user interaction to minimize the possibility of error.